ARMY RESEARCH LABORATORY

Finding a Minimum Covering Circle Based on Infinity Norms

by Andrew A. Thompson

ARL-TR-4495 **July 2008**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-4495 **July 2008**

Finding a Minimum Covering Circle Based on Infinity Norms

Andrew A. Thompson
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2008	Final	October 2007–February 2008

4. TITLE AND SUBTITLE
Finding a Minimum Covering Circle Based on Infinity Norms

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Andrew A. Thompson

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-WM-BF
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-4495

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report discusses the use of infinity norms to solve the problem of finding the minimum covering radius for a set of points. The minimum covering radius can be used as a measure of the accuracy of a collection of shots or a description of spall fragments. The algorithm worked well for the data sets investigated, sometimes converging in three iterations; however, in some cases, there were hundreds of iterations. For specific metrics, it would be possible to use directional derivatives to improve the convergence of the process. The overall design is based on defining an improvement step to be repeated until the state of the process fulfills a specific criterion. Infinity norms offer a theoretic framework for algorithm development.

15. SUBJECT TERMS
covering circle, infinity norm, algorithm

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Andrew A. Thompson
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	20	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6805

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

1. Introduction **1**

2. Method **1**

3. Implementation **3**

4. Conclusion **5**

Appendix. MATLAB Program Listing **7**

Distribution List **13**

List of Figures

Figure 1. Graphics user interface for covering circle algorithm. 3
Figure 2. Main function pseudo-code. 4
Figure 3. Function MoveCenter pseudo-code. 4
Figure 4. An inappropriate covering circle. 5

1. Introduction

This report discusses the use of infinity norms to solve the problem of finding the minimum covering radius for a set of points. The minimum covering radius can be used as a measure of the accuracy of a collection of shots or a description of spall fragments. The infinity norm differs from the 1-norm and the 2-norm in that it is determined by a single element of a set. The infinity norm is the largest deviation from a reference function or point, the 1-norm is the average deviation (always positive) from the reference function, and the 2-norm is the average of the square root of the square of the difference between the set of interest and a reference set. The 2-norm has been extensively used as it is commensurate with mathematical analysis. The 1-norm has become more popular as computing has made iterative methods more accessible.

An appointment of infinity norms is that they allow nonprobabilistic statements to be made. These statements are typically related to a performance metric of some type and have the following style: If the parameters are in this range, then the performance is better than this value. Notice that this form of statement is typical of mathematics and is the type of statement required for many artificial intelligence database applications; that is, it fits the framework of a predicate calculus. In this report, the infinity norm is used to make descriptive statements of the form: All the data is within a circle of radius r, or equivalently, if the data is from set A, then it is all within a circle with a radius of r. The method developed can be used for any n-dimensional space for any performance metric.

2. Method

A method to find the minimum covering radius for a set of points was developed within the framework of infinity norms. In this case, the set of controllable parameters is the center of a circle. The task is to adjust the center in a manner as to reduce the radius of the circle. The infinity norm is the greatest distance between each point and the center. A fine way to think of the algorithm is as an adjustable center removal process. For an infinity norm, typically only one point in the data set is pertinent for a given set of conditions. On a given iteration only the point associated with the infinity norm and the set of parameters are pertinent to reducing the magnitude of the norm. The gradient of the distance function for a particular point is toward the given point. For the minimum radius coverage problem, this amounts to moving the center towards the point associated with the infinity norm. The question then is how to determine the magnitude of this displacement. The problem is simple; it is not desirable to move so far that another point defines the infinity norm and has a larger magnitude.

As a first case, consider two points and assume the center is collocated at one of the points. Assuming distance is being used as the metric; one point will have a norm equal to the distance between the two and the other will have a norm of zero. If the magnitude of the difference in the norms is used as an adjustment for the center, the process will oscillate between the two points and never converge. This oscillation occurs along the direction connecting the two points, and is eliminated by using 0.5 as a multiplier of the magnitude. In this situation, when the midpoint between the two points is reached, there are two points with the same norm. Any change to the circles center from the midpoint can be thought of as a component along the connecting segment and a component orthogonal to this segment. Any change along the connecting segment will increase the infinity norm as one distance (metric) must increase. Changes in the orthogonal component will cause both distances (metrics) to increase. For distances, the Pythagorean Theorem guarantees this increase; for other metrics, the Cauchy Schwartz inequality can be invoked. So a deviation from the midpoint along either component predicts an increase of the infinity norm. This suggests an exit condition for an algorithm can be: If the center point is at the midpoint of two points and the norms associated with all the other points are less than these two norms, then the search is done.

Next, consider an exit condition for a set of three points, if the third point is within the circle defined by the two most distant points the situation reverts to the previous case; however, the third point can keep the center from reaching the midpoint of the two most distant points. If the center is located so that all three points are on the circumference of the same circle then the algorithm should exit as there is no basis for improvement in reducing the infinity norm. Consider the situation where the algorithm has found two points that have the same infinity norm. At this point for the distance metric, the center of the circle must be on the perpendicular bisector of the two points. To decrease the norms associated with these two points, the center must move toward their midpoint; during this process, it is possible for a third point to prevent further decrease when it obtains the same value as the other two points. In this situation, the center will be at the intersection of the perpendicular bisectors.

Based on the former discussion, an algorithm to find the minimum covering radius was developed. The basic step is to move towards the point with the largest distance from the center. This process is repeated until the difference between the two largest distances is within tolerances defined by the difference in the distances between the second and third largest distances. If the criteria is met, the process moves toward the midpoint of the segment defined by the two points with the largest distances. If the midpoint is reached, the algorithm terminates; if three points have equal distances (the same norm), then the process terminates. The algorithm uses a sort routine to order the distance metric and then a routine to find the index of a point with a given value. Both of these functions are available as MATLAB commands.

3. Implementation

The ideas were implemented through a graphical user interface (GUI) in MATLAB shown in figure 1. There are three buttons on the GUI. The first, called "Get Data," queries the user for a data file. The file is assumed to contain two rows of entries for the X and Y values of the data. If the data is stored from MATLAB, it must be in the first element using the save command. For example, if the data is in a variable named dat, then the command, "save info dat," will place the information in the variable dat into the file, "info.mat." After the file is opened, the GUI will display a graph of the data. To continue, the user must specify the max number of replications to be made. There is a slide bar to adjust this value; alternatively, the user can enter a number in the text box above the slide bar. After the maximum number of replications is set, the user then clicks the "Adjust Center" button. The ensuing graph shows the movement of the center. The number of iterations to meet termination requirements is displayed in the replications text box. The user can click the "Final Graph" button to see the end product. Displayed on the graph will be the data points in blue, the center of the circle as a red plus, and the circle in green. On the GUI under the title "Center," the center of the circle will be displayed; under the title "Radius," will be the minimum covering radius. Finally, if the user wants a figure for reports without the GUI interface, the menu item "AxestoFigure" can be clicked. This will open a figure window with the final graph using the file name as the title and the center location and radius printed below the x-axis. Properties can be adjusted using the figure menu item axes properties (under the edit menu). After this, the figure can be copied and moved to any document. The MATLAB program listing is included in the appendix.

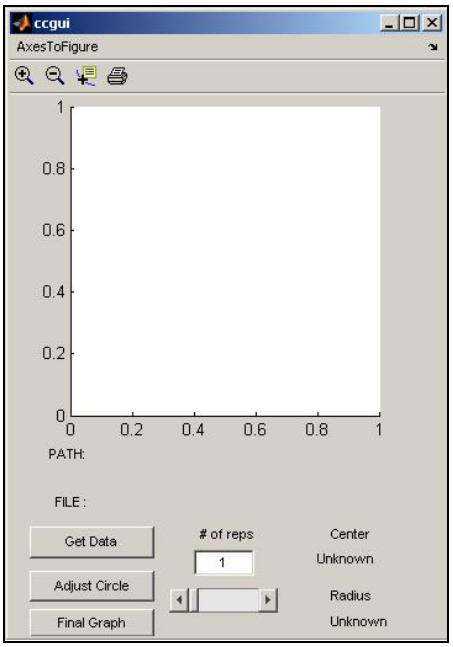

Figure 1. Graphics user interface for covering circle algorithm.

The starting point for the center was set to the mean of the locations. This worked for all the data sets investigated. The pseudo-code is contained in figures 2 and 3.

```
Main Function
Data is an array of points
Choose Center as Average of Data
lastCenter=Center+2*tolerance
For i=1 till # of iterations
        If distance between consecutive centers < tolerance
                Done
        Else
                Call MoveCenter
        Endif
Endfor
Done
```

Figure 2. Main function pseudo-code.

```
Function MoveCenter
//comment costfunction evaluates the data based on the metric
//comment sort puts the data in descending order high to low
Newdata=data-center
D2=costfunction(Newdata)
Ds=sort(D2)
If Ds(1)=Ds(2)=Ds(3)
        Done
Endif
```

Figure 3. Function MoveCenter pseudo-code.

It is prudent to mention the following situation as an example of the algorithm converging to an improper solution. Consider three points on a circle within a small arc. If the algorithm selects the center of this circle on any iteration it will terminate. This is an undesirable situation since the distance between the points will all be less than the radius of this circle; this is illustrated in figure 4. Although this situation did not occur using the mean of the locations as the starting point for the center, it did occur in one testing situation. Observation of the graph of the solution can be used to rule out this case. To test a solution it would be possible to restart the algorithm at the average of the three points defining the final circle and then determine if the algorithm converges to the same solution.

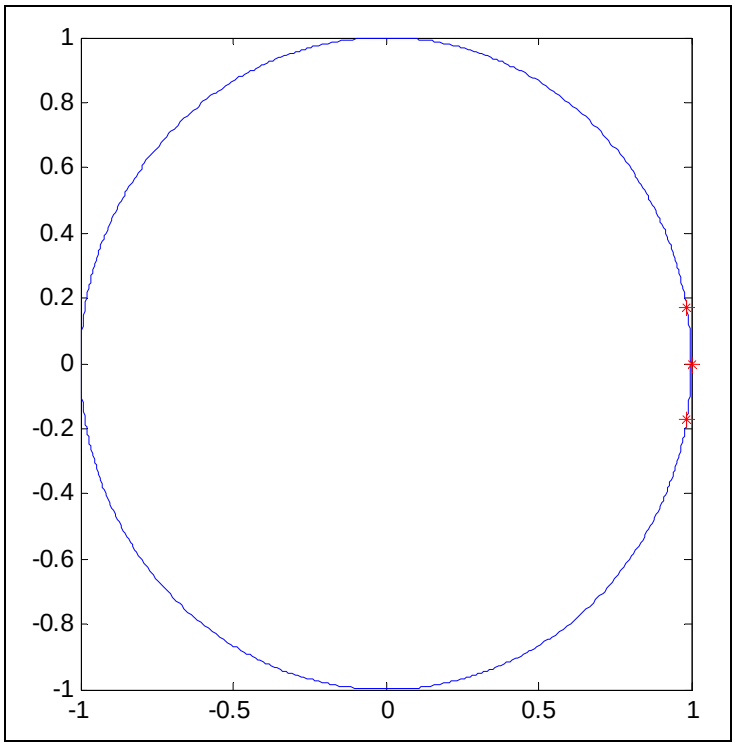

Figure 4. An inappropriate covering circle.

4. Conclusion

It is certainly possible to develop a more complex algorithm utilizing more detailed information. For example, the distant matrix could be calculated for the set of n points; this will take $0.5n^2 - n$ distance calculations. The Elzinga-Hearn method[1] is a geometric algorithm that solves this problem. The method developed is conceptually straightforward as it is based on the definition of the infinity norm. Unlike methods associated with linear programming, this method does not require linear metrics. The algorithm worked well for the data sets investigated, sometimes converging in three iterations; however, in some cases, there were hundreds of iterations. For specific metrics, it would be possible to use directional derivatives to improve the convergence of the process. The overall design is based on defining an improvement step to be repeated until the state of the process fulfills a specific criterion. Infinity norms offer a theoretic framework for algorithm development.

[1]Elzinga, D. J.; Hearn, D. W. Geometrical Solutions for Some Minimax Location Problems. *Transportation Science* **1972**, *6*, 379–394.

INTENTIONALLY LEFT BLANK.

Appendix. MATLAB Program Listing

This appendix appears in its original form, without editorial change.

```
function varargout = ccgui(varargin)
% CCGUI M-file for ccgui.fig
%      CCGUI, by itself, creates a new CCGUI or raises the existing
%      singleton*.
%
%      H = CCGUI returns the handle to a new CCGUI or the handle to
%      the existing singleton*.
%
%      This function finds the radius of a the covering circle of a set
%      of points (x,y)
%      The button get data is used to open a file containing the data
%      The button adjust data is used in conjunction with the number of
%      iterations to cycle in an attempt to find the center of the circle
%      with the smallest covering radius
%
%
%      CCGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CCGUI.M with the given input arguments.
%
%      CCGUI('Property','Value',...) creates a new CCGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ccgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ccgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ccgui

% Last Modified by GUIDE v2.5 04-Feb-2008 11:32:02
%Andrew Thompson
%2008 USARL
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ccgui_OpeningFcn, ...
                   'gui_OutputFcn',  @ccgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ccgui is made visible.
function ccgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ccgui (see VARARGIN)

% Choose default command line output for ccgui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ccgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ccgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushfile.
function pushfile_Callback(hObject, eventdata, handles)
% hObject    handle to pushfile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,path]=uigetfile('*.*');
if isequal(filename,0)
    disp('no change')
else
    set(handles.textpath,'string',path);
    set(handles.textfile,'string',filename);
    dat=load(fullfile(path,filename));
    if isstruct(dat)   %if saved as a *mat it is loaded as a structure take the first
field as data
       fnames=fieldnames(dat);
        dat=dat.(fnames{1});
    end
    [m,n]=size(dat);
    if n==2
        dat=dat';
    end
    axes(handles.axes1);
    hold off
    plot(dat(1,:),dat(2,:),'b*')
    set(handles.pushfile,'UserData',dat);
    %set(handles.textCenter,'string',' ');
    %set(handles.textRadius,'string',' ');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
set(handles.edit2,'string',num2str(floor(get(handles.slider2,'Value'))));

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Executes on button press in pbAdjustCircle.
function pbAdjustCircle_Callback(hObject, eventdata, handles)
% hObject    handle to pbAdjustCircle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dat=get(handles.pushfile,'UserData');
n=str2num(get(handles.edit2,'string'));
c=[mean(dat(1,:));mean(dat(2,:))];
clist=c;
for i=1:n
    clast=c;
    [c,r]=moveCenter(dat,c);
    cdif=c-clast;
    ncdif=norm(cdif);
    if ncdif<.00000001
        break;
    end
    clist=[clist, c];
end
set(handles.edit2,'string',num2str(i));
axes(handles.axes1);
plot(clist(1,:),clist(2,:),'b-',clist(1,:),clist(2,:),'r.',c(1),c(2),'r+')
set(handles.textCenter,'string',[num2str(c(1),4),', ',num2str(c(2),4)]);
set(handles.textRadius,'string',num2str(r,5));
cr=[c;r];
set(handles.pbAdjustCircle,'UserData',cr);



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dat=get(handles.pushfile,'UserData');
cr=get(handles.pbAdjustCircle,'UserData');
theta=0:.01:2*pi+.01;
axes(handles.axes1);
plot(cr(3)*cos(theta)+cr(1),cr(3)*sin(theta)+cr(2),'g')
hold on
plot(dat(1,:),dat(2,:),'b*')
plot(cr(1),cr(2),'r+')


% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_3_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_4_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_5_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function uitoggletool3_ClickedCallback(hObject, eventdata, handles)
```

```
% hObject    handle to uitoggletool3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function File_menu_Callback(hObject, eventdata, handles)
% hObject    handle to File_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
newfigurehandle = figure
copyobj(handles.axes1,newfigurehandle)
t=get(handles.textfile,'string');
t=['File: ',t];
c=get(handles.textCenter,'string');
xa=[' Center: (',c, ' )'];
r=get(handles.textRadius,'string');
xa=[xa, '    Radius: ',r];
title(t)
xlabel(xa)
print(newfigurehandle,'-dmeta')
%delete(newfigurehandle)



function [center,r]=moveCenter(dat,center)
%dat is a 2 dimensional set of data
%c is the assumed center
%
%the algorithm works by moving the center toward the
%data point with the greatest distance it is done when
%two or three points are all at the same approximate distance
%from the center
%this code is just one step so it needs to be called by a routine that
%decides when to stop adjusting

%Andrew Thompson
%basically an infinity norm minimization routine


[r,c]=size(dat);
if c==2
    dat=dat';
    n=r;
else
    n=c;
end

[r,c]=size(center);
if c==2
    center=center';
end

z=repmat(center,1,n);
newdat=dat-z;
ndat2=newdat.^2;
d2=sum(ndat2);
d2=sqrt(d2);
ds=sort(d2,2,'descend');
delta=.5*(ds(1)-ds(2));   %.5 minimizes the convergence oscillation
i=find(d2==max(d2));
%more than two points on circumpherence so done
if length(i)>2
    r=ds(1);
    return
end
%if length(i)==2
if ((ds(1)-ds(2)) < .2*(ds(2)-ds(3)) )
    if length(i) < 2
        i=[i find(d2==ds(2))];
    end
    mid=(dat(:,i(1))+dat(:,i(2)))/2;
    if center==mid
        %out='centered'
        r=ds(1);
        return;
    end
    dc1=mid-center;
    %dc=dc1/norm(dc1);
    delta=ds(1)-ds(3);
```

```
    if delta>norm(dc1)
        delta=1;
    end
    center=center+delta*dc1;
else
    dc=dat(:,i)-center;
    dc=dc/norm(dc);
    center=center+delta*dc;
end
z=repmat(center,1,n);
newdat=dat-z;
ndat2=newdat.^2;
d2=sum(ndat2);
d2=sqrt(d2);
ds=sort(d2,2,'descend');
r=ds(1);
```

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK T 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES ORGANIZATION

1 DREXEL UNIV
DEPT OF MECHL ENGRG
B CHANG
3141 CHESTNUT ST
PHILADELPHIA PA 19104

1 DIRECTOR
US ARMY RSRCH LAB
AMSRD ARL CI NT
R PRESSLEY
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

2 COMMANDER
USAATC
TEDT AT ADR
B GILLICH
S CLARK
400 COLLERAN RD
TRAILER T1
APG MD 21005-5059

1 COMMANDER
USAATC
TEDT AT ADF
S NOVAK
400 COLLERAN RD
APG MD 21005-5059

21 DIR USARL
AMSRD AAR AEF T
M ANDRIOLO
AMSRD ARL CI CT
B BODT
R KASTE
AMSRD ARL SL BD
J COLLINS
L MOSS
AMSRD ARL SL BW
P GILLICH
AMSRD ARL WM BA
R MCGEE
T BROWN
T HARKINS
M ILG
AMSRD ARL WM BF
J WALD
J WALL
D WEBB

NO. OF COPIES ORGANIZATION

M ARTHUR
A THOMPSON (4 CPS)
B FLANDERS
R PEARSON
B OBERLE